UNIVERSIDAD CARLOS II DE MADRID
ING. TÉCNICA EN INFORMÁTICA DE GESTION - PROYECTO FIN DE CARRERA

Simpleface:

Desarrollo y evaluación de una librería para la interacción ubicua.

Autor: Guillermo Tesoro Calvo
Tutor: Andrea Bellucci

Leganés, junio de 2014

Abstract.

In the context of ubiquitous computing, prototype designing is needing the integration of more and more disciplines. It is difficult to find a professional who has all the needed skills. This project pretends to provide an approximation of the problem. Therefore, the development of a software library is proposed, which allows a transparent and modular access to ubiquitous computing devices. The performance (in terms of reduction of development time) of this tool has been tested through an experimental study. The results obtained can be used to know the viability of systems that enable people of different areas to develop ubiquitous computing prototypes.

Resumen.

En el contexto de computación ubicua, el diseño de prototipos está necesitando cada día de más disciplinas. Es difícil encontrar a un profesional que aúne todas las competencias necesarias. Este proyecto pretende proporcionar una aproximación a este problema. Para ello, se plantea el desarrollo de una librería software que permita el acceso de forma transparente y modular a dispositivos de interfaz ubicua. La eficacia (en términos de reducción de los tiempos de desarrollo) de la herramienta se ha probado mediante una prueba experimental. Estos resultados sirven para decidir la viabilidad de sistemas que permitan a usuarios con diferentes perfiles involucrarse en el desarrollo de prototipos computación ubicua.

Tabla de contenido

1 Introducción.

1.1 Contexto.

Desde hace varias décadas, las computadoras han ido incorporándose a nuestra vida diaria de forma exponencial, hasta tal punto que hoy en día están implicadas en la mayoría de actividades humanas. Al mismo tiempo que las computadoras se iban incorporando a nuestras vidas, también cambiaba la forma de interactuar con ellas. En sus inicios, cuando tenían forma de grandes armarios y ocupaban habitaciones enteras, solo técnicos cualificados podían comunicarse con ellas mediante pilotos luminosos, tarjetas perforadas y pantallas de fosforo verde. A medida que la computación se extendía a otras tareas, fuera de las actividades bancarias o de investigación científica, también cambiaba la manera en que el usuario interactuaba con la máquina. Esto fue debido a que el perfil del usuario estaba cambiando. Ya no solo manipulaban ordenadores los operarios de las grandes computadoras del mundo de la banca, si no que oficinistas, administrativos, personal docente, etc, también incorporaron estas máquinas en sus trabajos. Teclados y monitores más avanzados fueron solo los primeros pasos en la tarea de facilitar la interacción entre humano y máquina. Acompañando los cambios en el hardware, aparecieron también avances en el software, traducidos como nuevas interfaces de usuario. Los procesos batch, lotes de instrucciones donde se detallaba todo el comportamiento que debía tener el computador, dieron paso a las interfaces basadas en línea de comando, donde el usuario podía comunicarse con el computador mediante comandos preestablecidos. Estos comandos indicaban a la maquina lo que el usuario deseaba hacer, creándose así un especie de dialogo primitivo entre ambos. Con la llegada del ratón aparecieron las interfaces graficas de usuario. Estas interfaces hacían uso del ratón como dispositivo de apuntado y utilizaban metáforas del mundo real (ej. Ventanas, escritorio, etc) para representar y organizar la información que fluía entre el usuario y la computadora.

La aparición de las interfaces gráficas propició el desarrollo de nuevos dispositivos de interfaz. Este nuevo hardware fue usado para explorar nuevas metáforas para la interacción entre el humano y la computadora, intentando acercar las interfaces de la maquina a las costumbres humanas (tocar, hablar, mover, agarrar, etc). De esta forma, con prototipos como el Sketchpad [Jonhson 1963], que sentó las bases del diseño asistido por computador (CAD), las interfaces comenzaron a adaptarse al ser humano y no a la inversa, como ocurrió en los primeros tiempos de la computación. A este lo siguieron diversos avances como las pantallas táctiles, que permitían interactuar directamente con lo que vemos en la pantalla utilizando únicamente nuestros dedos, los sistemas de visión computarizada, que permiten a la

computadora tomar y analizar imágenes del mundo real para tomar decisiones, lo que permite, por ejemplo, un control gestual de las interfaces. Estos avances dieron lugar un gran abanico de nuevas posibilidades a la hora de crear computadores. Estas nuevas ideas abandonaban el concepto arcaico de computadores mastodónticos para incorporarlos a las rutinas cotidianas de todo el mundo. Mark Weiser denominó esta idea computación ubicua [Weiser 1991, 1993; Satyanarayanan 2001; Saha and Mukherjee 2003]. Este término define un tipo de computación en el que los ordenadores dejan de ser herramientas en sí mismas y se integran a nuestro propio entorno de una forma transparente. Por consiguiente, al formar parte de nuestro entorno, la interacción ha de ser necesariamente natural para que no percibamos a los computadores como un elemento aislado. Ya que la computación ubicua es un concepto nuevo que además, diverge de la imagen clásica de computador, también requieren de nuevos conceptos para comunicarnos con ellos, nuevas interfaces.

Han sido ya muchos los pasos dados en esta dirección. Por ejemplo, las interfaces tangibles, donde el usuario interactúa con el entorno digital por medio de objetos físicos de su entorno, las interfaces touchless, donde los gestos del usuario son interpretados por un sistema de visión computarizada o por sensores de movimiento como en el caso del mando de la consola Nintendo WiiMote[1]. Estos avances han sido potenciados por la reducción de costes debida a nuevas tecnologías y mejoras en los procesos de producción. Dos buenos ejemplos son las pantallas táctiles [Jefferson, 2005], que hoy en día están prácticamente en cada aparato electrónico, o Kinect[2], un dispositivo desarrollado por Microsoft que incorpora una cámara infrarroja para detectar profundidad y otra RGB para conseguir la imagen real. Hace solo algunos años, estos sistemas eran poco accesibles debido a su precio y baja disponibilidad, sin embargo en la actualidad, cualquiera puede desarrollar sus ideas haciendo uso de ellos.

Este auge en la disponibilidad ha facilitado la investigación en el área de la computación ubicua, y más concretamente en el desarrollo de nuevos prototipos de interfaces naturales. Estos nuevos dispositivos, debido a su idiosincrasia, han saltado del terreno puramente informático, en el sentido más clásico, para convertirse en objetos cotidianos, lo que implica la necesidad de un diseño horizontal, donde varias disciplinas se unen a la informática para crear nuevos dispositivos. Es en este donde comienzan a surgir las problemáticas derivadas de la multidisciplinaridad.

[1] http://en.wikipedia.org/wiki/Wii_Remote
[2] http://www.microsoft.com/en-us/kinectforwindows/

1.2 Problema.

Como mencionan Myers et al. [2000], las interfaces de usuario han evolucionado hasta requerir la intervención multidisciplinar en su desarrollo. Hay pocos profesionales que aúnen todos estos conocimientos lo que nos lleva a la necesidad de que cada experto tenga unos conocimientos mínimos en los campos que le son ajenos. De esta forma, el ingeniero informático ha de conocer ciertos conceptos de diseño o psicología mientras que el diseñador ha de tener ciertas habilidades técnicas en el campo de la computación, etc.

Este hecho difumina las barreras existentes entre carreras profesionales y crea nuevas profesiones que aglutinen conocimientos de varias áreas. Dos ejemplos de los últimos tiempos son los ingenieros de sonido o multimedia. Ambos tienen unos conocimientos sólidos en técnicas de computación o electrónica pero a su vez extienden su formación más allá en áreas como la música o las artes gráficas respectivamente.

Esta tendencia nos lleva a un contexto donde cada vez más gente, con diferentes niveles de conocimiento técnico y diferentes áreas, investiga sobre nuevas formas de interacción y posibilidades de las interfaces ya existentes. Este hecho en sí mismo ya plantea un problema. Hay muchos profesionales ajenos a la informática con buenas ideas que ayudarían a mejorar las interfaces existentes, e incluso a crear nuevas, haciendo uso de los conocimientos de sus áreas de origen (Diseñadores, psicólogos, sociólogos, artistas, etc). En principio, todo son ventajas, muchos especialistas aportando sus ideas para mejorar las experiencias de interacción con ordenadores, pero no toda esta gente puede trabajar en un entorno multidisciplinar donde cada profesional aporta sus ideas y estas son llevadas a cabo en conjunto. En la mayor parte de los casos, se trata de investigadores aficionados, que emplean su tiempo libre y sus propios medios en desarrollar este tipo de ideas.

Desde el comienzo de la informática personal, incluso antes, estos entusiastas se han organizado en torno a diferentes movimientos socioculturales. Desde el movimiento phreaker[3], germen del actual movimiento hacker, pasando por la creación en los años 70 del Homebrew Computer Club[4], el cual sería una pieza clave en el estallido de la informática personal, hasta la cultura Maker[5] más reciente. Cada nueva movimiento cultural ha ido expandiendo las ideas básicas de la filosofía DIY (Do it yourself) y aplicándolas a diferentes áreas.
En el caso de la computación ubicua es aún más importante si cabe aplicar los conceptos DIY. Los prototipos en el desarrollo de dispositivos de computación

[3] http://en.wikipedia.org/wiki/Phreaking
[4] http://en.wikipedia.org/wiki/Homebrew_Computer_Club
[5] http://en.wikipedia.org/wiki/Maker_culture

ubicua son esenciales para entender el alcance y las posibilidades del proyecto. Podemos destacar el auge de las plataformas de prototipado rápido como Arduino[6] o Raspberry Pi [7]. Estas plataformas nacieron como proyectos abiertos a la comunidad y han permitido a entusiastas de todo el mundo desarrollar conceptos de computación ubicua de una forma sencilla en formato de prototipo.

El prototipado rápido es una colección de técnicas de diseño, desarrollo y construcción que permiten la creación de prototipos de forma rápida y directa, proporcionando así un modelo en el cual estudiar todas las posibilidades del dispositivo, especialmente aquellas que no han sido predichas en el diseño inicial. Según Klemmer [2012], estos prototipos deben seguir tres directivas.

- No debe ser completo
- Debe ser fácil de modificar
- Puede ser dejado de lado cuando no sea necesario

En los entornos multidisciplinares, estos prototipos pueden ser usados como el centro de una tormenta de ideas, donde cada participante puede dar puntos de vista relacionados con su área de conocimiento, dando así un enfoque multidisciplinar al proceso de diseño dirigido por prototipos.

Teniendo en mente la idea del prototipado, y recordando que no todos estos aficionados o investigadores pueden desarrollar sus ideas en entornos donde exista un intercambio de conocimiento entre diferentes áreas, aún quedan problemas relativos a los conocimientos necesarios para desarrollar ideas. Pongamos un ejemplo. Un diseñador gráfico desea crear un proyecto de interacción que hace uso de acelerómetros. El WiiMote es la herramienta idónea, ya que es barata y accesible, pero sus conocimientos en desarrollo software no son suficientes para integrarlo en el desarrollo de su prototipo. Debido a su formación como diseñador gráfico, el área del diseño de la interfaz de su prototipo no supone un problema, pero carece de los conocimientos técnicos necesarios para integrar los acelerómetros. Podría intentar buscar esta información en internet, pero en muchos casos esta está fragmentada y carente de estructura, lo que puede acabar por abrumar al diseñador y hacerle abandonar la idea. Lo que nos lleva a la definición del problema.
En el área del diseño de dispositivos de computación ubicua, el incremento en los conocimientos necesarios relativos a diferentes áreas, impide que una sola persona pueda desarrollar prototipos, lo que nos conduce a una pérdida de ideas que nos consiguen traspasar la barrera técnica necesaria.

[6] http://www.arduino.cc/
[7] http://www.raspberrypi.org/

1.3 Solución propuesta y objetivo.

Teniendo en cuenta la problemática anteriormente descrita, esto es, la elevada dificultad técnica que conlleva el desarrollo de prototipos para personas ajenas a la parte más técnica de la informática, se propone la creación de una herramienta que reduzca la cantidad de conocimientos relativos al software en la creación de prototipos de computación ubicua. Concretamente, se ha creado una prueba de concepto en formato de librería, que proporcione funcionalidades de alto nivel relativas a dispositivos de interacción al usuario final.
Como objetivo final del proyecto, se comprobará si la librería supone un beneficio a la hora de desarrollar prototipos interacción ubicua.

1.4 Contribuciones.

En el proceso necesario para conseguir el objetivo propuesto, se han generado las siguientes contribuciones.

- Los requisitos necesarios para el desarrollo de la librería. En todo proyecto software, se ha de comenzar por la definición de los requisitos con respecto a lo que se espera del desarrollo. En este caso, los requisitos han de estar centrados en facilitar el desarrollo de prototipos de computación ubicua y la sencillez de cara al usuario final.
- La creación de un diseño genérico, jerárquico y modular. La librería, a pesar de tener carácter de prueba de concepto, ha de posibilitar su posible ampliación y ser independiente de los protocolos de comunicación que usen los dispositivos que abstrae. Para ellos, se creara un diseño en capas que mantenga una alta modularidad, para que cualquier paso entre el dispositivo y la API de usuario final pueda ser intercambiable.
- Una prueba experimental que comprobará si la librería supone efectos positivos en la eficiencia a la hora de desarrollar prototipos de interacción ubicua

1.5 Sinopsis

En esta memoria se tratarán los siguientes apartados.

- **Estado del arte**: Resumen y análisis de las tecnologías actuales relacionados con el trabajo que se va a hacer.
- **Análisis**: Análisis técnico del problema y elección de herramientas y procedimientos.
- **Diseño**: Diseño de la parte software del proyecto.
- Implementación: Descripción técnica sobre la implementación del diseño especificado.
- **Evaluación**: Descripción de la prueba con la que se evaluará la herramienta.
- **Resultados**: Resumen de los resultados obtenidos en la evaluación.
- **Interpretación estadística**: Interpretación de los resultados obtenidos en la prueba por medio de herramientas estadísticas y conclusiones cualitativas.
- **Gestión del proyecto**: Resumen de la gestión del proyecto.
- **Conclusiones y trabajos futuros**: Análisis de las conclusiones obtenidas durante el desarrollo del proyecto y posibles continuaciones del mismo.

2 Estado del Arte.

Una vez definido el contexto y antes de comenzar el desarrollo del proyecto, es necesario realizar un análisis del estado tecnológico de la computación ubicua y los diferentes periféricos de interfaz, ya que estos serán los pilares a la hora de desarrollar el proyecto. Una vez evaluadas las tecnologías existentes, se estudiara la posibilidad de hacer uso de ellas, o bien, desarrollar soluciones nuevas si no existe ninguna que se adapte a las necesidades del proyecto.

En esta memoria se introducirá el concepto de computación ubicua de manera formal. Con esta base, se analizarán las principales tecnologías de interacción ubicua existentes, tales como touchless, multi-touch y tangibles, y sus herramientas de desarrollo. Además, se hablara del movimiento Maker y el prototipado rápido y su implicación en la computación ubicua haciendo un análisis de sus principales herramientas.

2.1 Computación ubicua.

Para poder entender el marco del proyecto es necesario conocer el concepto de computación ubicua. Este ya ha sido definido brevemente, sin embargo, en esta sección se definirá el concepto con mayor exactitud y se listarán las principales tecnologías que giran en torno a la computación ubicua.

A principios de los noventa, Weiser definió formalmente la computación ubicua basándose en sus trabajos anteriores en el Palo Alto Research Center (PARC) de Xerox [Weiser, 1991]. La idea se puede resumir en la siguiente frase: “Maquinas que encajan en el entorno humano en lugar de forzar a este a entrar en el entorno de las mismas” [York and Pendhakar, 2004]. Fue el propio Weiser quien, al verse ante un concepto tan amplio, intuyo la necesidad de incorporar psicólogos y sociólogos a la tarea del diseño de computadores ya que este nuevo tipo de computación requiere el entendimiento de las circunstancias sociales, psicológicas y culturales de tu ámbito para poder funcionar de una manera óptima. Al fin y al cabo, este concepto tiene un origen humanista al intentar adaptar una tecnología a nuestros hábitos y no al contrario.

Debido a lo amplio del concepto, Weiser no dio tanto una definición concreta del término si no una taxonomía donde clasificar los diversos dispositivos y tecnologías. Para Weiser, pueden ser clasificados en tres grupos.

- Tabs: Dispositivos del orden de los centímetros diseñados para llevar encima, interconectados y con capacidad de procesamiento propio. Ejemplos: Smartphones, GPS, etc.

- Pads: Dispositivos del orden de los decímetros diseñados para manipularse y ser sostenidos con las manos, sin necesidad de ser apoyados en ninguna superficie (Al contrario que los ordenadores portátiles). Weiser los definió como "pedazos" de ordenador, análogamente a los pedazos de papel donde se hacen anotaciones, como una cuartilla. Ejemplos: iPad y tabletas similares.

- Boards: Dispositivos del orden del metro diseñados para servir como visores de información interactivos dentro del mobiliario del hogar o la oficina. Ejemplos: Microsoft Surface.

Debido a lo laxo del concepto y la taxonomía de la computación ubicua, muchas han sido las tecnologías que han nacido de esta idea, incluso, diversos nombres para referirse a ella que aporta sutiles matices como computación pervasiva, "internet de las cosas" o computación háptica.

Estas son algunos dispositivos actuales que han desarrollado los conceptos de Weiser aplicados a la computación ubicua.

- **Perceptive Pixel.**[8]
  Basado en la tecnología Multi-touch [Han and Davidson, 2006] (Pantallas táctiles con capacidad de reconocer dos o más toques simultáneos), esta compañía creo en 2006 un sistema que hace uso de gestos naturales para manipular de forma contextual lo que hay en pantalla.

- **Microsoft Surface.**[9]
  Tomando conceptos de la tecnología de Jefferson Han, Microsoft creó una mesa que ampliaba sus capacidades de interacción mediante el reconocimiento de objetos que se ponían en su superficie.

- **Microsoft Kinect.**
  Kinect es una combinación de cámaras RGB e infrarroja que permite detectar la profundidad, de forma que habilita la detección en tiempo real de objetos reales o personas.

[8] http://www.microsoft.com/office/perceptivepixel/
[9] http://www.microsoft.com/en-us/pixelsense/default.aspx

- **Sixth Sense.**[10]
  Este prototipo creado por Pranav Mistry y presentado en el SIGGRAPH de 2009 [Mistry et al, 2009], une un picoproyector, una cámara y unas pegatinas para los dedos para crear interfaces contextuales manipuladas por gestos.

2.2 Herramientas para computación ubicua.

Dado que el objetivo del proyecto es la creación de una librería para el desarrollo de prototipos de computación ubicua, se debe hacer un análisis de las tecnologías existentes en la actualidad.

En el contexto de la computación ubicua, actualmente podemos hablar de diferentes escenarios de interacción en el espacio de diseño. En este texto se trataran herramientas para el desarrollo de diferentes interfaces, tales como touchless, multi-touch y tangibles.

2.2.1 Touchless

Se llama interfaz touchless a aquellas que están basadas en sensores y no requieren de una interacción directa con las pantallas donde se muestra la información. Actualmente no existe ningún estándar o paradigma para este tipo de interfaces, pero su interés está creciendo dentro de la industria y el entorno académico. Debido a que no existe una única forma de crear interfaces touchless, se listarán herramientas de desarrollo describiendo el método para el que están orientadas.

Hardware	Nombre	Lenguajes
Webcams	Microsoft Touchless	C#
Wiimote, Kinect	GlovePIE	Scripting
Hardware que cumpla el estándar OpenNI	OpenNI	C++/Java
Kinect	Microsoft Kinect SDK	C++/C#

[10] http://www.pranavmistry.com/projects/sixthsense/

- **Microsoft Touchless[11] - Webcam - C#**
  Microsoft Touchless es una librería de código abierto que permite crear interfaces utilizando el reconocimiento de marcadores mediante una webcam. Estos marcadores son distinguidos del entorno por su color, lo cual hace fácil el desarrollo de prototipos pero a su vez limita la eficacia de la librería para ir más allá.
- **GlovePIE[12] - Kinect/WiiMote - Scripting**
  Glove Programmable Input Emulator, o GlovePIE, es un emulador de entradas clásicas para ordenador. Usando su propio lenguaje de scrtipting, podemos mapear acciones de Kinect o Wiimote como entradas emuladas de teclado o ratón, lo cual es también su principal limitación.
- **OpenNI[13] – Hardware conforme con OpenNI – C++/Java**
  OpenNI es un framework para interacción natural de código abierto, desarrollado por un consorcio industrial sin ánimo de lucro liderado por PrimeSense. Da soporte a diferentes tipos de hardware tales como cámaras de infrarrojos (Kinect), arrays de micrófonos, etc. Su diseño esta basado en middlewares que permiten ampliar las funcionalidades del framework utilizando diferente hardware. Sus principales limitaciones son el hecho de funcionar solo con una colección limitada de hardware, así como estar orientado a perfil técnico alto.
- **Microsoft Kinect SDK[14] – Kinect – C++/C#**
  Como su propio nombre indica, el Microsoft Kinect SDK es una librería desarrollado para la creación de aplicaciones que hagan uso de Kinect. Esta librería permite sacar partido del sensor de profundidad de Kinect y su array de micrófonos. Sus funcionalidades son similares a las de OpenNI, pero debido a que se trata de una librería y no de un framework, no proporciona ningún tipo de arquitectura y se presenta con un conjunto de funcionalidades limitado. Además, igual que OpenNI, está orientado a usuarios con un perfil técnico alto.

[11] http://touchless.codeplex.com/
[12] http://glovepie.org/glovepie.php
[13] http://openni.org
[14] http://www.microsoft.com/en-us/kinectforwindows

2.2.2 Multi-touch

El termino multi-touch se refiere a cualquier tipo de tecnología que nos permita interactuar con superficies utilizando más de punto de toque a la vez. Esto incluye no solo el movimiento de varios puntos de toque sobre las superficies, sino también los gestos que podemos realizar con los dedos (Pellizcos, deslizamientos, etc). Dependiendo de la tecnología utilizada a nivel de hardware, pueden ser detectados dos o más eventos al mismo tiempo. Las herramientas para desarrollar aplicaciones que hagan uso de interfaces multi-touch pueden estar orientadas a uno o más tecnologías, lo cual condiciona sus funcionalidades.

TUIO.[15]
A pesar de no ser una librería ni un framework, TUIO es un estándar de intercambio de datos para gestos multi-touch, así como objetos tangibles (que serán tratados más adelante). TUIO es un estándar abierto para la comunicación de los sistemas de detección. Utiliza un protocolo cliente-servidor que estandariza la comunicación entre el controlador y las aplicaciones que subyacen al mismo. Se tomará como punto positivo el soporte a TUIO, debido a que estandariza las comunicaciones, lo que nos permite usar la herramienta junto a otros servicios o dispositivos que sean compatibles con el estándar.

Nombre	Lenguajes	TUIO
Touchlib	C++	Sí
libavg	C++/Python	Sí
MutiTouch for Java (Mt4j)	Java	Sí

- **Touchlib**[16] **– C++**
  Touchlib es una librería software para el desarrollo de aplicaciones multi-touch compatible con diferentes técnicas de detección (DSI, FTIR y DSI). Además, también proporciona soporte para interacción tangible pudiendo identificar objetos físicos por medio de marcadores. Su funcionamiento está basado en el procesamiento de imágenes de video en crudo haciendo uso de librerías como OpenCV para la detección y seguimiento de objetos luminiscentes (blobs). Además es compatible con el protocolo TUIO. Aunque oficialmente solo funciona en Windows, existen una implementación cross-platform llamada Comunity Core Vision. CCV es una de los entornos de desarrollo más completos a la hora de desarrollar aplicaciones multi-touch, lo que también conlleva una alta complejidad y hace necesario un perfil técnico alto.

[15] http://www.tuio.org/
[16] http://nuigroup.com/touchlib/

- **Libavg[17] – C++/Python.**
  Libavg es una librería versátil para el procesamiento de audio, video e imágenes. Incluye soporte para capturar datos directamente desde videocámaras así como la posibilidad de utilizar diferentes filtros, incluyendo algunos acelerados por GPU. Contiene funciones para la detección y seguimiento de cuerpos dentro de la imagen. Debido a sus características, libavg no puede usarse de forma directa, ya que solo aporta funciones para el procesamiento que han de ser usadas para la construcción de un sistema de reconocimiento multi-touch. Debido a esto, requiere un alto nivel de conocimientos técnicos.
- **MutiTouch for Java (Mt4j)[18] – Java**
  Multitouch for java es una framework de código abierto construido en Java que da soporte a diferentes sistemas de entrada, buscando la posibilidad de permitir el reconocimiento de eventos multi-touch. Tiene una arquitectura diseñada en capas que permite el uso de diferentes periféricos (Dispositivos Multi-touch, ratones, teclados, eventos de red, etc), de forma que se pueden dar las mismas funcionalidades utilizando diferentes dispositivos. La abstracción hardware, junto con la compatibilidad con TUIO, son sus grandes ventajas con respecto a otros frameworks.

2.2.3 Tangible

La idea principal de las interfaces tangibles es el uso de objetos del mundo real para interaccionar con los sistemas. Esto se puede llevar a cabo de diferentes formas, por ejemplo, estableciendo una relación 1:1 entre los objetos reales y sus contrapartidas virtuales. Estas interfaces, al igual aquellas basadas en técnicas multi-touch, se basan en las mismas técnicas de detección de eventos, luego la mayor parte de las librerías que podemos encontrar para la interacción tangible se basan en librerías multi-touch ampliadas para el conocimiento de objetos. Debido a estás similitudes, también se valorara el soporte para TUIO.

Nombre	Lenguajes	TUIO
Papier-Mâché	Java	No
reacTIVision	C++	Sí
libTISCH	C++ (C#, Java, Python)	Sí

[17] https://www.libavg.de/
[18] www.mt4j.org

- **Papier-Mâché [19]– Java.**
  Papier-Mâché es una colección de herramientas que da soporte a la interacción con objetos físicos por medio de etiquetas de radiofrecuencia y códigos de barras. Es uno de los primeros kits de herramientas que unen detección por visión con detección electrónica (RF). Debido a esto, su principal baza es la de abstraer al programador de lidiar con cámaras y detectores de RF. Además proporciona una interfaz de sencilla de cara a la manipulación de objetos físicos. No soporta TUIO.

- **reacTIVision[20] – C++.**
  reacTIVision es un framework para sistemas de detección de objetos físicos desarrollado por los creadores de la Reactable. Se trata de un framework de código abierto que usa técnicas de visión por computador orientadas a la detección de objetos reales, y su manipulación como dispositivos de interfaz. Está especialmente orientado al reconocimiento de marcadores en superficies, para lo cual usa un algoritmo propio. Además de esto, proporciona una alta compatibilidad con el estándar TUIO, hasta el punto de incluir un simulador llamado TUIOSimulator para hacer pruebas sistémicas.

- **libTISCH[21] – C++ (C#, Java, Python).**
  libTISCH es una framework que propone una arquitectura por capas para el la detección y presentación de objetos para el desarrollo de aplicaciones de interfaz tangible. Sus diferentes capas funcionan de forma independiente comunicándose entre ellas por medio de un protocolo privado. El framework cubre toda la línea de procesamiento, desde la detección del objeto, a la presentación en pantalla por medio de librerías gráficas. Tiene soporte para TUIO y unos requisitos de conocimientos técnicos altos.

[19] http://hci.stanford.edu/research/papier-mache/
[20] http://reactivision.sourceforge.net/
[21] http://tisch.sourceforge.net/

2.3 Movimiento Maker.

En el capítulo anterior se han citado y analizado varias herramientas para la creación de software para la computación ubicua, pero, debido a la alta imbricación que tiene este tipo de computación con su contrapartida hardware se ha de hablar del movimiento Maker. Ya se mencionó en la introducción pero en esta sección se tratara de la implicación del movimiento Maker en el avance de la computación ubicua.

La cultura o movimiento Maker, nace de las bases del concepto Do It Yourself. Las comunidades humanas han estado largo tiempo practicando el "hágalo usted mismo", debido a que solo podían confiar en sí mismo para fabricar o reparar las herramientas del día a día. Con la llegada de la producción en masa y el aumento de la complejidad en los útiles se ha dejado de lado esa idea para priorizar la compra por encima de la reparación o la construcción. Kuznetsov y Paulos. [2010] definen el DIY como la creación, modificación o reparación de un objeto si ayuda profesional. En los últimos años, con la llegada de la sociedad de la información, amateurs de todo el mundo comparten grandes cantidades de conocimiento a través de la red, que en la mayoría de los casos han sido adquiridos de forma experimental. Esto ha motivado el resurgimiento de movimientos DIY en diferentes campos, especialmente en el caso del hardware y las tecnologías de la información. Esto nos lleva al movimiento Maker, que lleva el DIY a la fabricación de objetos y herramientas, particularmente en el contexto de la computación y la robótica, pero sin limitarse a estos campos. Por ejemplo, Instructables[22] es un portal web que se define a sí mismo como "Plataforma de documentación en la web donde gente apasionada puede compartir que hacen y como lo hacen". La web contiene proyectos DIY en formato paso-a-paso, donde los usuarios pueden comentar y valorar cada tutorial, aportando así una componente social. El proyecto no está limitado a la creación de hardware, y en ella podemos encontrar información desde cómo construir un guantelete medieval a controlar una célula GSM usando un Arduino.

2.4 Prototipado rápido

Como ya se mencionó anteriormente, el prototipado rápido constituye una parte clave en el desarrollo de dispositivos de computación ubicua. Esto nos permite desarrollar rápidamente una idea y entender mediante una manipulación practica el alcance de la misma.
El prototipado rápido, a través de una exploración reflexiva sobre el prototipo, nos permite confirmar o descartar las hipótesis sobre las que se construye la idea que se desarrolla, además de estimar posibles funcionalidades que no han sido previstas. Cuando los sistemas que se quieren diseñar crecen en complejidad, un prototipo

[22] http://www.instructables.com/

rápido ayuda a calibrar el alcance y corregir de forma temprana desviaciones sobre los objetivos que se quieren alcanzar. En resumen, el prototipado rápido es una buena estrategia para lidiar con cosas que no son fáciles de predecir, lo cual sucede especialmente cuando el campo de trabajo aun está en desarrollo, como puede ser el de la computación ubicua.

Cuando hablamos en términos de interacción con sistemas informáticos, la idea de prototipado ha evolucionado desde una herramienta para la evaluación de una idea, hasta llegar a convertirse en un proceso que permite a los diseñadores reflejar y comunicar una idea durante el diseño [Lim et al., 2008]. Según Buxton [2007] este proceso de prototipado ha de venir acompañado con el de bosquejar. Crear bocetos, junto con los prototipos, representa un hilo conductor fundamental en la innovación aplicada a la interacción y por ende a la computación ubicua.

A la hora de crear dispositivos de computación ubicua, aparecen nuevos grados de libertad en contraposición con el diseño clásico de dispositivos de computación, los cuales están ligados al escritorio en la mayoría de los casos. La computación ubicua permite más libertad a la hora de diseñar, ya que se no se tiene que restringir a los dispositivos de entrada clásicos, como el ratón o el teclado, sin embargo, esto incrementa las dificultades conceptuales y de diseño. A mayor libertad, mas variables en juego lo que conlleva un incremento en la complejidad. En la mayoría de casos, esto se traduce, como se ha venido hablando a lo largo de este documento, en una necesidad mayor de dominar diferentes áreas de conocimiento tales como, programación, electrónica, procesado de señal, etc. Esta dificultad se hereda a la hora de realizar prototipos rápidos y durante los últimos años diversas herramientas y frameworks han intentado subsanarlo. A continuación se repasaran algunas de ellas.

2.4.1 Herramientas para el prototipado rápido en la computación ubicua.

- **OpenInterface[23] - C++ / Java /MATLAB / .NET**
  Desarrollado por un consorcio de asociaciones académicas, investigadoras y corporativas, OpenInterface propone un sistema visual para el desarrollo de sistemas multimodales donde puedan interactuar diferentes dispositivos. Utiliza una arquitectura de tuberías y filtros, donde cada filtro crea o procesa un flujo de datos que después se transfiere al siguiente mediante una o más tuberías. Esto permite la creación de sistemas tanto sencillos como complejos. Lamentablemente, su última versión, de 2010, ya no está disponible para descarga y ha sido imposible probarla.

[23] http://www.openinterface.org/home/

- **Squidy[24] - Java**
  Squidy, al igual que OpenInterface, es un sistema de programación visual basado en componentes modulares. Utiliza tres componentes de alto nivel, Squidy manager, Squidy designer y Squidy Client. Su sistema funciona mediante nodos, los cuales permiten la abstracción para su uso por parte de personas con bajos conocimientos de programación, pero a la vez, mediante un sistema de zoom, permite modificar el código fuente de cada nodo en caso de ser necesario.

- **ROSS - Java**
  Responsive Objects, Surfaces, and Spaces, o ROSS, es uno de los mayores esfuerzos en lo a que interacción ubicua se refiere. Define una arquitectura anidada donde se definen entidades físicas como espacios, superficies y objetos. Un espacio puede contener superficies o objetos, los cuales a su vez pueden contener superficies de nuevo. Se define también un cuarto tipo para designar controles físicos como botones o diales. La comunicación entre los diferentes elementos se realiza mediante XMLs haciendo uso de una red de elementos. Su principal ventaja es que su API no está definida como funcionalidades de cada elemento sino como interacciones de los mismos. Esto a su vez limita su alcance y posibilidades.

2.4.2 Plataformas hardware para el prototipado rápido.

Una vez hemos analizado las posibles librerías y herramientas que nos permiten crear prototipos de computación ubicua, debemos hablar también de las posibles plataformas para el desarrollo de nuevos dispositivos.

- **Arduino**
  Arduino es una plataforma de hardware libre que proporciona una placa base, un microcontrolador y un entorno de desarrollo para el mismo. Está diseñado para facilitar la creación de pequeños proyectos hardware de una forma sencilla, gracias a su IDE basado en Processing, el cual permite un aumento de la abstracción sobre el clásico VHDL a la hora de programar microcontroladores. Desde su creación, ha tenido la voluntad de acercar el prototipado electrónico a artistas, diseñadores y aficionados, y como tal ha evolucionado el proyecto, priorizando la sencillez y manteniendo un compromiso con la versatilidad. Actualmente existe una gran comunidad alrededor de la plataforma, lo que facilita el encontrar materiales, tutoriales, etc.

[24] http://hci.uni-konstanz.de/index.php?a=research&b=projects&c=16386645

- **Intel Galileo**[25]
  Siguiendo la estela de popularidad creada por Arduino, en 2013 la compañía Intel saco al mercado la placa Intel Galileo (compatible con el ecosistema Arduino), añadiendo al concepto de microcontrolador de propósito general, un procesador x86, lo que permite el desarrollo de prototipos más autónomos y que requieran un mayor poder de procesamiento, el cual queda lejos del alcance de un microcontrolador.

- **Raspberry Pi**
  Nacido como un proyecto de carácter social, Raspberry Pi fue concebido como un proyecto que permitiera llevar a países en vías de desarrollo un ordenador completo y de muy bajo coste. El diseño final de la placa incorpora un SoC Broadcom de arquitectura ARM, junto con 256MB de RAM lo convierten en un ordenador completo. Posee salidas HDMI y puertos USB para poder convertirlo en un ordenador de sobremesa, además de incorporar una interfaz GPIO, lo que permite a la Raspberry en una plataforma de prototipado de gran potencia y bajo coste. Actualmente su uso se ha extendido por toda la comunidad Maker trascendiendo así de su objeto humanitario.

[25] http://www.intel.es/content/www/es/es/do-it-yourself/galileo-maker-quark-board.html

2.5 Conclusiones

Después del análisis realizado sobre las herramientas de desarrollo para interacción y computación ubicua, así como plataformas de prototipado rápido, podemos llegar a las siguientes conclusiones.

- La mayoría de las herramientas requieren unos conocimientos técnicos altos, lo cual eleva la barrera de entrada para empezar a desarrollar.
- Las herramientas están orientadas a un tipo concreto de dispositivo de entrada, o en el mejor de los casos, a un conjunto que comparta características, lo que reduce las posibilidades de interacción a nivel de librería.
- Ofrecen un nivel muy bajo de abstracción, lo cual las convierte en herramientas potentes pero complejas de usar.

Estos puntos, unidos a lo comentado ya en la sección de introducción y a lo largo del texto, nos llevan a pensar en el desarrollo de una librería que minimice la barrera de conocimientos necesarios y ofrezca una abstracción clara sobre los dispositivos. Además deberá ser modular, lo que fomente la interacción de diferentes tipos de dispositivos en las capas intermedias con la intención de ofrecer funcionalidades de más alto nivel.

3 Análisis.

El análisis de un proyecto debe servir de guía para la posterior etapa de diseño. Esta sección se dividirá entre casos de uso, que definirán las diferentes formas en las que la herramienta será utilizada, y requisitos, donde se definirán de forma simple y directa las funcionalidades que deberá cubrir el sistema, sin entrar en detalles técnicos.

3.1 Casos de uso

Tal y como se ha descrito, existen dos usos diametralmente diferentes de la herramienta. Por un lado se definirá el escenario del desarrollador cuya función es ampliar la librería con nuevos dispositivos y funcionalidades. Por otro lado, tenemos el rol de usuario, que utilizará las nuevas funciones creadas por el desarrollador en sus propios prototipos.

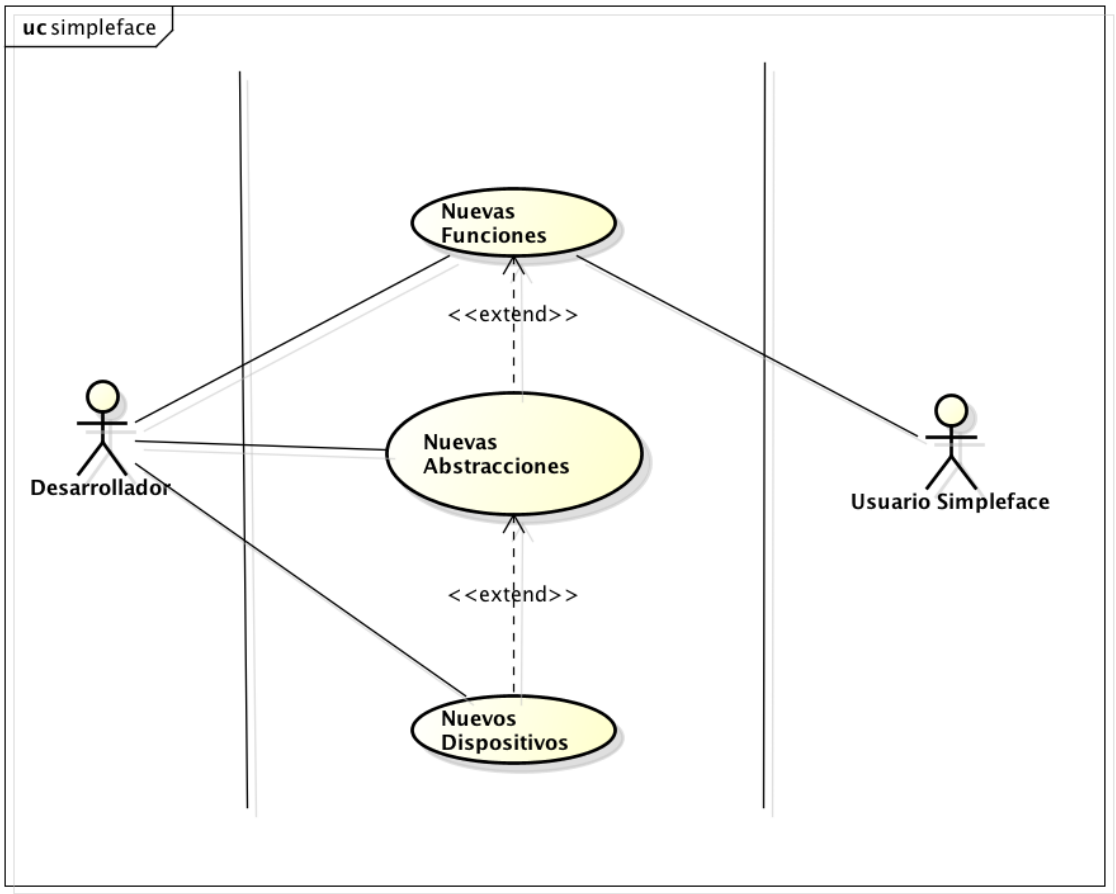


Ilustración 1 Diagrama de Casos de Uso Simpleface

3.1.1 Descripción textual.

A continuación se detallará el flujo de trabajo de cada caso de uso, incluyendo los actores involucrados y las condiciones necesarias.

UC-01	
Nombre	Nuevos dispositivos
Actores	Desarrollador
Objetivo	Implementar un nuevo dispositivo en la biblioteca
Precondiciones	Ninguna
Postcondiciones	La librería soportará un nuevo tipo de dispositivo
Escenario Básico	1. El desarrollador creara o buscará un driver de bajo nivel para el nivel para el nuevo dispositivo. 2. Se creara una abstracción mínima para el driver usando funciones atómicas

Tabla 1 Caso de uso 1

UC-02	
Nombre	Nuevas abstracciones
Actores	Desarrollador
Objetivo	Implementar un nuevo conjuntos de abstracciones
Precondiciones	Al menos un dispositivo en la librería
Postcondiciones	La librería soportará un nuevo tipo de abstracciones
Escenario Básico	1. El desarrollador elegirá uno o varios dispositivos ya presentes en la librería. 2. Se creara un nuevo conjunto de funciones a partir de funciones atómicas de uno o varios dispositivos

Tabla 2 Caso de uso 2

UC-03	
Nombre	Nuevas Funciones
Actores	Desarrollador
Objetivo	Implementar un nuevo conjuntos de funciones de alto nivel
Precondiciones	Al menos un conjunto de abstracciones en la librería
Postcondiciones	La librería soportará un nuevo conjunto de funciones de alto nivel
Escenario Básico	1. El desarrollador elegirá una o varias abstracciones ya presentes en la librería. 2. Se creara un nuevo conjunto de funciones de alto nivel a partir de un conjunto de una o unas abstracciones

Tabla 3 Caso de uso 3

UC-04	
Nombre	Nuevas Funciones
Actores	Usuario Simpleface
Objetivo	Utilizar funciones de alto nivel para el desarrollo de aplicaciones
Precondiciones	Al menos un conjunto de funciones de alto nivel en la librería
Postcondiciones	la aplicación del usuario incluirá funciones de alto nivel proporcionadas por la librería
Escenario Básico	1. El usuario elegirá uno o varios conjuntos de funciones relacionadas con uno o varios dispositivos. 2. El usuario usara las funciones proporcionadas por la librerías en el desarrollo de su aplicación.

Tabla 4 Caso de uso 4

3.2 Requisitos

Mediante la definición de requisitos, se detallarán una serie de funcionalidades y restricciones que definirán el proceso de diseño. Estos requisitos ayudan a que el diseño se ciña a la creación de una solución para la problemática planteada y no añada funcionalidades innecesarias o contenga interrelaciones o efectos secundarios no tipificados en el análisis del problema.

Normalmente, estos requisitos se definen en reuniones o entrevistas con el cliente que requiere el software a desarrollar. En este caso, al ser el proyecto tanto un desarrollo como un estudio sobre la prueba de concepto, no existe la figura del cliente, luego los requisitos deberán ser definidos por el experimentador en dos sentidos. Primero, que el desarrollo tenga las funcionalidades necesarias para resolver la problemática definida en un inicio, y segundo, que la prueba de concepto desarrollada durante el proyecto, este orientada al desarrollo de un estudio comparativo. De esta forma, los requisitos se dividirán en dos. Requisitos de Software y Requisitos de evaluación.

3.2.1 Requisitos de software.

SR-01	
Nombre	Modularidad
Descripción	La librería deberá ser fácilmente ampliable por medio de módulos en todos los niveles de abstracción para facilitar la ampliación.

SR-02	
Nombre	Interfaces sencillas
Descripción	La librería deberá utilizar interfaces sencillas en todos sus niveles, tanto entre los módulos de la librería como la interfaz externa con el usuario. La premisa siempre será mantener la sencillez de uso de cara al usuario de la librería.

SR-03	
Nombre	Priorizar sencillez contra versatilidad
Descripción	El conjunto de funciones ofrecidas al usuario final deberá ser sencillo y con usos concretos. Este precepto ha de priorizarse contra la versatilidad de las funciones.

SR-04	
Nombre	Independencia
Descripción	El desarrollo de la librería deberá encapsular los todas las herramientas externas, tanto drivers como terceras librerías. Todas las herramientas de terceros deberán ser transparentes al usuario final.

SR-05	
Nombre	Processing
Descripción	El desarrollo de la librería deberá ser realizado para la plataforma Processing. Esto implica seguir sus estándares de creación de las mismas.

SR-06	
Nombre	Instalación mediante el estándar de Processing
Descripción	La librería deberá poder ser instalada en el entorno Processing sin requerir de ninguna acción adicional por parte del usuario, siguiendo el procedimiento estándar.

Tabla 5 Requisitos de Software

3.2.2 Requisitos de Evaluación

ER-01	
Nombre	Dispositivos de estudio
Descripción	Con el objetivo de establecer una prueba de concepto para la evaluación experimental, se desarrollará soporte para dos dispositivos en la librería. Arduino con ADXL335 y WiiMote (solo funcionalidad de acelerómetro).

ER-02	
Nombre	Hipótesis y evaluación.
Descripción	La definición de hipótesis a comprobar deberá ser concreta y contrastable. Así mismo, para evaluar los datos recabados durante la evaluación, deberá usarse un método estadístico contrastado y confiable.

ER-03	
Nombre	Diseño de pruebas
Descripción	Las pruebas que serán usadas en la evaluación de la herramienta deben tener objetivos simples que evalúen las características básicas de la librería. Incluyendo como mínimo el uso de interfaces y documentos. Estás pruebas deberán ser sencillas e implicar el mínimo numero de conocimientos adicionales posible.

ER-04	
Nombre	Sujetos de estudio
Descripción	Los sujetos de estudio deberán tener un nivel de conocimientos homogéneo sobre la materia (Programación, diseño, conocimientos sobre interfaces, etc).

Tabla 6 Requisitos de evaluación.

4 Diseño

Una vez han sido definidos los requisitos y los casos, se debe sintetizar toda esa información en un diseño que permita desarrollar un sistema adecuado al objetivo. Para este propósito, el primer paso es crear un diagrama de clases que describa la arquitectura elegida.

4.1 Arquitectura

Para seguir la especificación de los requisitos, el diseño de la arquitectura se ha centrado en la modularidad y la facilidad de ampliación del sistema. A continuación se muestra el diagrama de clase diseñado.

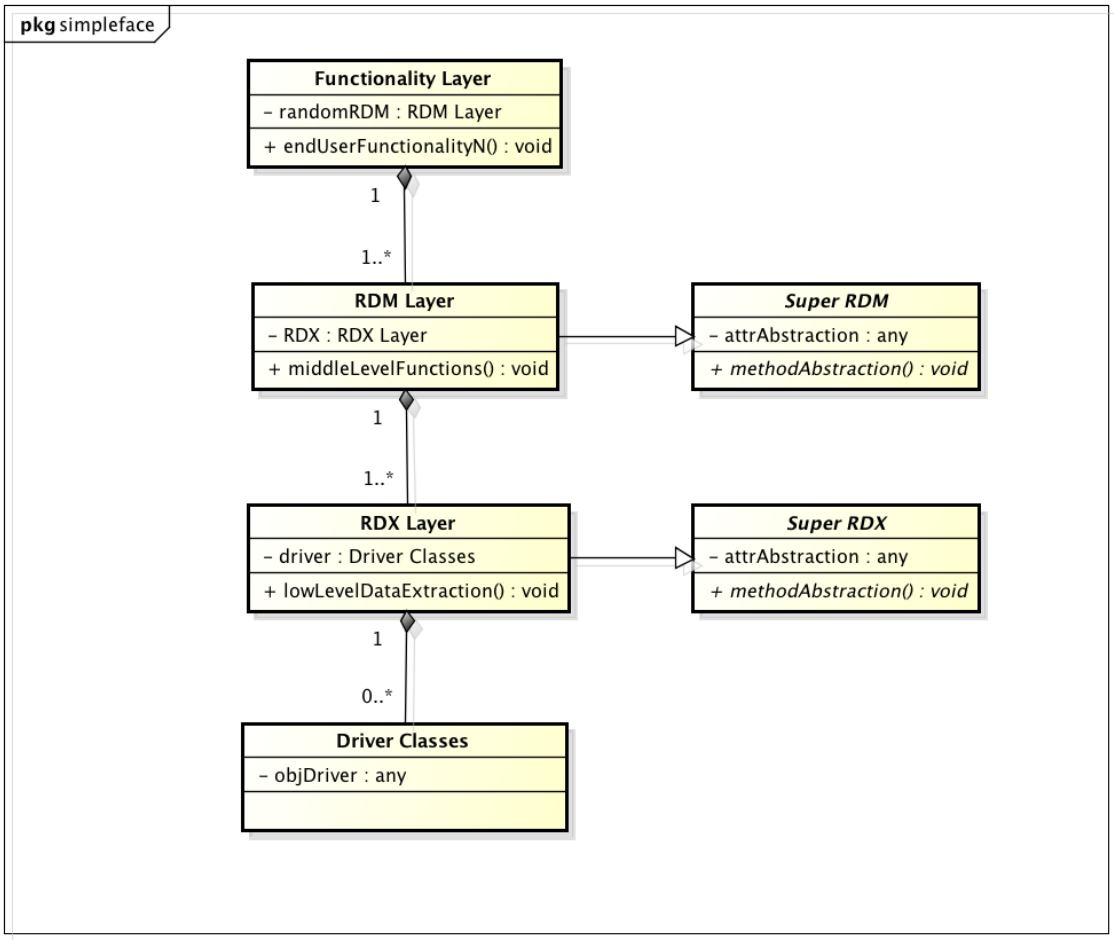


Ilustración 2 Diagrama de clase

En el diagrama se pueden observar tres niveles de abstracción, RDX, RDM y Functionality. Estos diferentes niveles proporcionan un escalado en la abstracción del sistema, lo que permite crear diferentes implementaciones en el mismo nivel totalmente intercambiables entre si, ya que las interfaces entre niveles están definidas por súper clases. De esta forma, se pueden definir súper clases para cada tipo de dispositivo (acelerómetros, mesas táctiles, punteros, etc) utilizando

diferentes implementaciones para cada tipo de hardware, manteniéndose la compatibilidad entre niveles. Cada dispositivo que se quiera implementar en la librería debería tener, al menos, una clase de cada capa para poder completar la pila de abstracción desde el driver hasta las funcionalidades de alto nivel.
A continuación se detallaran en profundidad los niveles de abstracción.

4.1.1 Capa RDX (Raw Data Extraction)

Esta es la capa de más bajo nivel del modelo y, por ello, es usada para encapsular el driver del dispositivo a implementar y extraer los datos que este proporciona. La capa RDX deberá implementar una interfaz con el driver del dispositivo, de esta forma, se crea transparencia desde este punto hacia arriba en la pila de abstracción.

La capa RDX deberá proporcionar funciones atómicas para emular las funcionalidades del driver, que deben estar definidas en la superclase RDX. Está superclase ha de definir las funcionalidades comunes a todos los dispositivos de una misma clase para permitir la transparencia entre diferentes hardware.

Por cada tipo de dispositivo que soporte la librería, existirá una súper clase RDX que establezca un interfaz común a todos ellos, consiguiendo con esto mantener la compatibilidad en la pila de abstracción. Esto es, dos dispositivos con una funcionalidad igual deberán implementar la misma súper clase RDX para ser soportados.

Los drivers usados por la capa RDX deberán estar preferiblemente integrados en la librería, para mantener el carácter cerrado y facilitar la distribución.

4.1.2 Capa RDM (Raw Data Management)

De la misma forma en la que la capa RDX proporciona abstracción sobre el driver, la razón de ser de la capa RDM es la de servir de abstracción a capa la capa RDX, usando las funciones atómicas proporcionadas por la misma y creando otras de mas alto nivel, con las cuales se crearan las funcionalidades de cara al usuario de la librería. Esta capa RDM puede integrar una o varias clases RDX, pudiendo así el soporte de dispositivos multifuncionales (ej: Dispositivo Apuntador + Acelerómetro), o de la misma forma, generando funcionalidades intermedias en las que intervengan dos dispositivos diferentes.
Las funcionalidades implementadas en la capa RDX han de ser lo suficientemente sencillas como para no limitar la creación de funcionalidades en la capa de Funcionalidad. Han de limitarse a dar un sentido semántico a los datos proporcionados por las clases RDX que implemente.
De la misma forma que con la capa RDX, se creará una súper clase RDM por cada tipo de dispositivo que soporte la librería, de esta forma podrán usarse implementaciones diferentes de capas RDM sin perturbar la compatibilidad con el resto de capas.

4.1.3 Capa Functionality

La capa Functionality es la última capa de abstracción, y la única visible por parte del usuario de la librería.
Esta capa debe implementar una o varias clases RDM para generar funcionalidades concretas relativas al dispositivo que este emulando. Igual que el resto de capas, puede implementar diferentes capas RDM para proporcionar la funcionalidad de un dispositivo complejo o simplemente encapsular un único RDM para dar una funcionalidad concreta y sencilla.
Está clase no tiene súper clase ya que no debe mantener la compatibilidad con ninguna otra, simplemente proporcionar una interfaz sencilla y concreta de comunicación con el dispositivo de interfaz de natural.
Las funciones que implemente deben ser los suficientemente concretas como para facilitar el uso del dispositivo, pero lo más sencillas que sea posible para proporcionar la máxima versatilidad al usuario.

Idealmente, todas las capas deben estar documentadas, pero se ha de tener especial cuidado con esta ya que es la parte visible de la librería.

5 Implementación

Una vez definido el diseño, se ha de proceder a la implementación. Ya que el desarrollo se ha creado como una prueba de concepto, se ha decidido implementar las funcionalidades de un acelerómetro. Para demostrar el diseño modular y la abstracción de hardware, se utilizaran dos tipos de hardware diferentes. Por un lado un WiiMote conectado con interfaz Bluetooth, y por otro una placa Arduino conectada a un acelerómetro ADLX335 conectado por medio de una interfaz serie a través de USB. Para implementar ambos dispositivos, se creara un estándar para las funcionalidades de acelerómetro, lo que permitirá el uso indistinto manteniendo las interfaces.

5.1 Estándar Simpleface: Acelerómetro.

A continuación se detalla el diseño de clases usado para la implementación de diferentes dispositivos acelerómetro , siguiendo el estándar de la librería Simpleface.

5.2 RDXWiimote

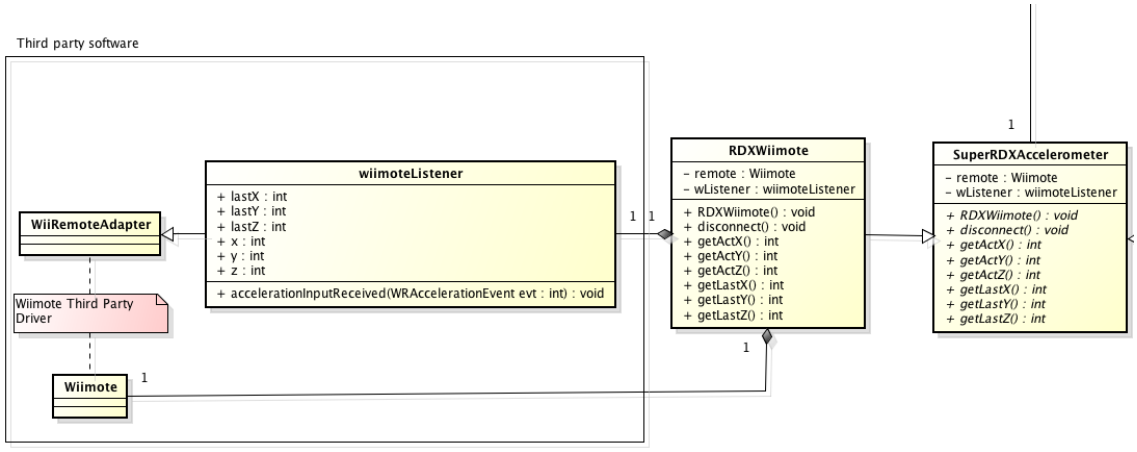


Ilustración 3 Diagrama RDX Wiimote

En este diagrama podemos ver como la clase RDX implementa un driver para Wiimote proporcionado por un tercero, en este caso concreto, WRJ4P5, que aparece recuadrado en el diagrama. Este driver usa una implementación libre del estándar bluetooth para la comunicación con el Wiimote que es distribuida con la librería.

La clase RDX extrae los a datos a través de la funcionalidad proporcionada por WRJ4P5, implementando así la súper clase RDX Accelerometer, que define los métodos básicos que han de implementar todos los dispositivos que proporcionen funcionalidad de acelerómetro.

5.3 RDXArduino

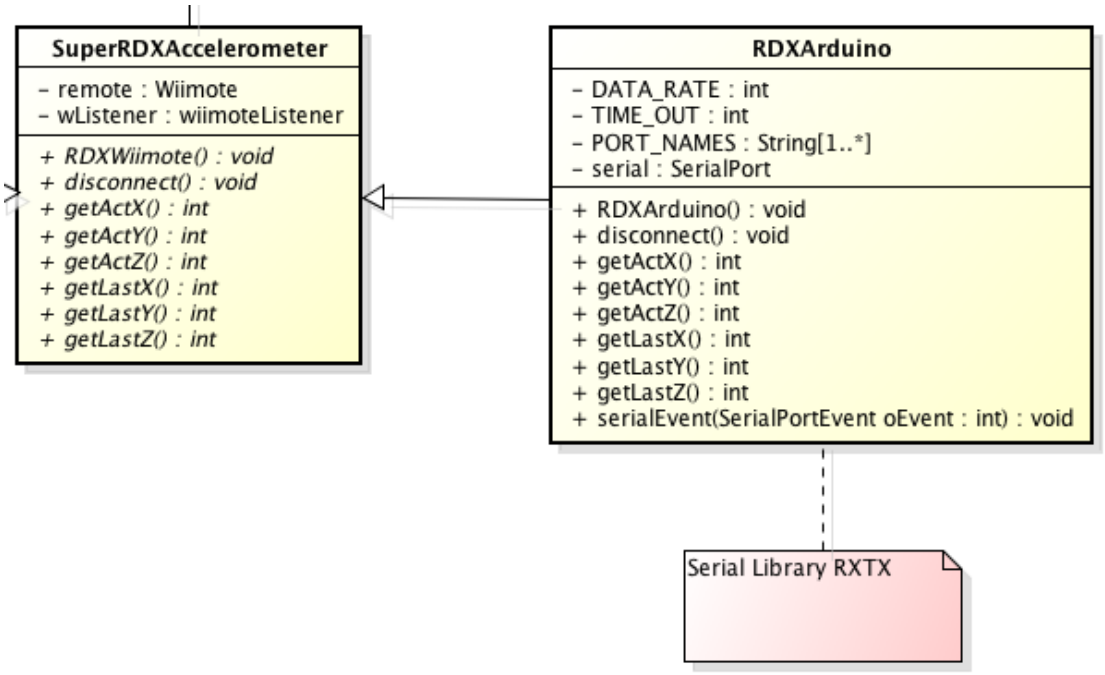


Ilustración 4 Diagrama RDX Arduino.

En este caso, se puede observar como el RDX de Arduino no implementa ningún driver. Esto es debido a que el Arduino usado para el proyecto tiene un firmware mínimo con un protocolo de comunicación sencillo a través de bus serie (será especificado más adelante), de forma que la clase solo necesita encapsular el uso de la librería RXTX de Java.

De la misma forma que RDX Wiimote, esta clase implementa todos los métodos de Super RDX Accelerometer para mantener la compatibilidad con las clases superiores de la pila de abstracción.

5.3.1 Protocolo de comunicación Arduino.

El Arduino usado en el proyecto utiliza un protocolo de comunicación muy simple basado en texto ASCII enviado a través del puerto serie a 9200 baudios.
La siguiente expresión regular define cada paquete:

[0..9]+X[0..9]+Y[0..9]+Z -> ej. 54X120Y30Z

El número que precede a cada letra representa el valor de la aceleración en ese mismo eje. Estos paquetes se envían constantemente a través del bus serie de forma que la letra Z indicara el fin de una trama y el comienzo de otra.

6 Evaluación

6.1 Definición del experimento y su contexto.

En el contexto de interacción ubicua, la tendencia es ampliar el número de diferentes inputs que el usuario es capaz de realizar, ya sea por vías más tradicionales, como el ratón y el teclado, o a través de nuevos tipos de sensores como superficies táctiles, acelerómetros, cámaras RGB e infrarrojas, etc. La herramienta Simpleface sirve como una capa de abstracción de los diferentes sensores. De esta forma se quiere facilitar el desarrollo rápido de prototipos de interfaces que hagan uso del hardware anteriormente citado. El objetivo es crear una experiencia de interacción más rica y acorde con la manera en la que los humanos interactuamos con nuestro entorno naturales.

Para llevar a cabo este estudio, se utilizarán 16 sujetos con un perfil único especificado más adelante en este documento. Estos sujetos deberán realizar un prototipo definido por el estudio utilizando tanto la herramienta Simpleface como métodos tradicionales con el fin de contrastar los mismos. Este estudio pretende recoger los datos paramétricos producidos durante la realización del prototipo por parte de los sujetos y sus resultados finales, así como estudiarlos cuantitativa y cualitativamente con el fin de determinar la veracidad de las hipótesis planteadas a continuación.

6.2 Hipótesis.

El experimento está orientado a comprobar si la herramienta Simpleface facilita la implementación de mecanismos de interacción por medio de sensores utilizando el entorno de Processing. Estos conceptos se pueden expresar a través de las siguientes hipótesis nulas:
H_{n1}: La herramienta Simpleface no afecta de manera positiva a la eficiencia (tiempo de desarrollo, cantidad de errores por línea) en el desarrollo de prototipos que hagan uso de dispositivos para la interacción natural.
H_{n2}: La herramienta Simpleface no minimiza la cantidad de conocimientos previos necesarios para la configuración del entorno desarrollo.

Hipótesis alternativas:
H_{a1}: La herramienta Simpleface afecta de manera positiva a la eficiencia (tiempo de desarrollo, cantidad de errores por línea) en el desarrollo de los prototipos que hagan uso de dispositivos para la interacción natural.
H_{a2}: La herramienta Simpleface minimiza la cantidad de conocimientos previos necesarios para la configuración del entorno desarrollo.

6.3 Variables

Variables independientes: Tecnología (Wiimote o Arduino), Entorno de desarrollo (sin librería o con librería Simpleface)
Variables dependientes: Tiempo de ejecución de la tarea, error de programación durante la prueba.

6.4 Perfil de los sujetos de estudio.

Los sujetos de estudio que participarán en el estudio forman parte de la Universidad Carlos III de Madrid y se pueden homogenizar bajo el perfil de estudiante, graduado de Ingeniería Informática que actualmente cursan un Master o realizan un doctorado en alguna de las áreas del campo de la interacción humano-maquina (HCI de aquí en adelante). Debido al área de especialización de estos estudiantes son los que mejor podrían encajar en el puesto de Interaction Designer , además, no es necesario que tengan conocimientos previos sobre la plataforma de desarrollo Processing, ya que debido a la formación previa, los sujetos tienen unos conocimientos de técnica informática que les permitiría comprender fácilmente el funcionamiento del entorno de desarrollo. Para poder afinar los resultados de cada uno de los sujetos, sus conocimientos previos serán evaluados en un test previo al estudio que se describirá a continuación.

6.5 Descripción de la encuesta preliminar y post-estudio

6.5.1 Preliminar

Para la correcta evaluación de los resultados que arrojen las implementaciones realizadas por los sujetos de estudio debe existir un conocimiento previo sobre los mismos. En la encuesta que se definirá continuación se pretende obtener información de dos tipos, biológica y técnica. La manera de obtener la información será mediante preguntas concretas con opciones de respuesta fija que pueden indicar valores concretos o puntuaciones en escala para evaluar competencias o rasgos.
Esta encuentra se encuentra en el APENDICE A.

6.5.2 Post-Estudio

Con el objetivo de obtener retroalimentación por parte del sujeto de estudio después de la realización de las tareas, se ha creado un test de evaluación de la experiencia del estudio. Este test ha de ser contestado con respecto a la experiencia con las tareas realizadas.
Esta encuentra se encuentra en el APENDICE B

6.6 Descripción de escenarios y tareas.

Para la evaluación de las dos hipótesis planteadas se han diseñado dos tareas, una asociada a cada hipótesis.

T_1: Conseguir información para configurar el entorno de desarrollo Processing (Con y sin Simpleface) para poder realizar prototipos usando un dispositivo acelerómetro (Wiimote o Arduino).
T_2: Programar una prueba de concepto que haga uso de un dispositivo acelerómetro (Wiimote o Arduino) utilizando la librería y sin hacer uso de ella.
El experimento seguirá la metodología de diseño factorial[26]. Cuando tenemos dos o más variables independientes, como es el caso (Con y sin Simpleface, Wiimote o Arduino) es lógico querer analizarlas en el mismo experimento. Para ello, todas las posibilidades de cada variable independiente se han de combinar con todas las posibilidades del resto de variables, lo que el caso que nos aplica se define con la siguiente terminología, 2x2, esto es, dos variables independientes con dos posibilidades cada una.
Para combinar las variables independientes en el diseño factorial se puede optar por dos opciones, within[27] y between[28] subjects. En el caso primero, se aplican todas las posibilidades a cada sujeto de estudio mientras que en el caso between subjects, se crean diferentes grupos y se aplica una posibilidad a cada uno de ellos. El escenario ideal es utilizar siempre within subjects para eliminar sesgos y tener unos resultados más homogéneos, pero a su vez hay que tener un compromiso con los tiempos de duración de las pruebas y el número de posibles combinaciones que se podrían dar.

Teniendo esto en cuenta, en las tareas diseñadas para el experimento se utilizó un diseño mixto within y between subjects (Split-Plot Design). La metodología between subjects se utilizará para dividir a los sujetos entre los dos dispositivos. La mitad de los sujetos realizarán las tareas utilizando un Wiimote y la mitad restante usará un Arduino. Por otro lado, la realización de la segunda tarea será llevada a cabo usando la metodología within subjects, ya que todos los sujetos realizaran la prueba con y sin librería. Para evitar que el efecto del aprendizaje sesgue la evaluación, se alternan las pruebas en los sujetos, primero con librería o primero sin librería. Este efecto será también reducido debido a un pequeño entrenamiento que será dado por igual a todos los sujetos del estudio donde se les enseñaran las bases de Processing y cómo funcionan los acelerómetros. En cualquier caso, debido a las grandes diferencias entre los dos entornos de desarrollo, no se esperan grandes desviaciones debidas al efecto del aprendizaje.

---

[26] http://psych.csufresno.edu/psy144/Content/Design/Experimental/factorial.html
[27] http://www.experiment-resources.com/within-subject-design.html
[28] http://www.experiment-resources.com/between-subjects-design.html

El conjunto de las tareas y el entrenamiento previo no durará más de 90 minutos (Nielsen 2005). Este tiempo se dividirá de la siguiente manera:

- 15 minutos para realización de encuesta pre-experimento y entrenamiento sobre Processing.
- 15 minutos para la búsqueda de información (T_1)
- 40 minutos para la prueba de programación (T_2).
- 10 minutos para la encuesta post-experimento y explicación de los objetivos.

6.7 Protocolo.

El desarrollo del experimento se realizará mediante sesiones individuales, en un entorno controlado. Se utilizará el protocolo Think Aloud [Ericsson y Simon, 1980] y las sesiones serán supervisadas por dos experimentadores. De los cuales solo uno interactuará con los sujetos (Contestar dudas, introducir las tareas, etc); esta decisión ha sido tomada para evitar sesgos introducidos por el cambio de experimentador (por ejemplo, diferentes formas de expresión).
El experimentador tomará nota de los tiempos de inicio y fin de cada tarea, del número de errores por línea de código en T_2 y por los comentarios surgidos a raíz del protocolo Think Aloud.

6.8 Fase de entrenamiento.

Durante los 15 minutos previos a la realización de la prueba, el experimentador impartirá un pequeño entrenamiento donde se explicara al sujeto las características básicas de la plataforma Processing necesarias para llevar a cabo la prueba. Este entrenamiento constará de los siguientes puntos:

- Introducción a la plataforma Processing,
- Concepto de Sketch
- Como importar librerías.
- “Hola Mundo”

En el caso de los sujetos que no conozcan previamente el funcionamiento de un acelerómetro, también se explicará brevemente el concepto.

6.9 Escenario y tareas.

Los textos a continuación serán entregados a cada sujeto para la realización de las pruebas. Sirven como texto para cada tarea y ayudan a contextualizar la situación.

6.9.1 Contexto

"Desde hace un año trabajas en una empresa dedicada a la investigación de nuevas interfaces de usuario y mejoras de usabilidad. La junta directiva de la empresa ha decidido crear un nuevo grupo de investigación compuesto por varios perfiles (diseñadores, programadores, artistas, etc) dedicado a crear prototipos de interacción. Tu eres uno de los integrantes de ese grupo perteneciente al perfil más técnico.
Se os ha pedido que investiguéis sobre posibles usos de diferentes dispositivos de entrada. En tu caso, el dispositivo asignado ha sido el acelerómetro. Unas pocas búsquedas en internet te clarifican el concepto de acelerómetro aplicado a la interacción con computadores. Se trata de un dispositivo electrónico que es capaz de medir las aceleraciones que se generan en cada eje (x,y,z) y devolverlas en un formato numérico.
Después de una breve recogida de datos a través de internet determinas que el dispositivo con acelerómetro más extendido es el Wiimote. Además, en otro de los departamentos de tu empresa han desarrollado un prototipo de acelerómetro basado en Arduino, de forma que decides utilizar estos dos dispositivos para comenzar el desarrollo de prototipos.
Con el objetivo de trabajar colaborativamente con tus compañeros, elegís el entorno de desarrollo Processing debido a su facilidad de uso."

6.9.2 Tarea de configuración (T_1)

"El primer paso para comenzar a desarrollar prototipos (Sketchs de Processing) es la configuración de un entorno. Para ello necesitarás encontrar información sobre cómo se realiza la conexión de los dispositivos (Arduino y Wiimote) con Processing.
Para el caso del Wiimote deberás encontrar un driver que te permita interactuar con el a través de Processing . Debido a que Processing es básicamente un entorno de desarrollo basado en Java, decides buscar un driver para Wiimote escrito para este lenguaje.
En el caso del prototipo de Arduino, la interfaz de comunicación es el puerto serie de forma que lo más sencillo es buscar una librería que te permita utilizar la comunicación serie desde Processing.
Por último, en una de las divisiones de tu empresa, se ha desarrollado un prototipo de librería dedicada a proporcionar abstracción sobre diferentes sensores, encontrándose entre ellos el acelerómetro. La división dedicada a su desarrollo te ha proporcionado toda la documentación necesaria sobre la librería, llamada Simpleface.

Esta tarea se dará por terminada cuando consideres que tienes suficiente información para poder empezar a programar el prototipo (Wiimote o Arduino).”

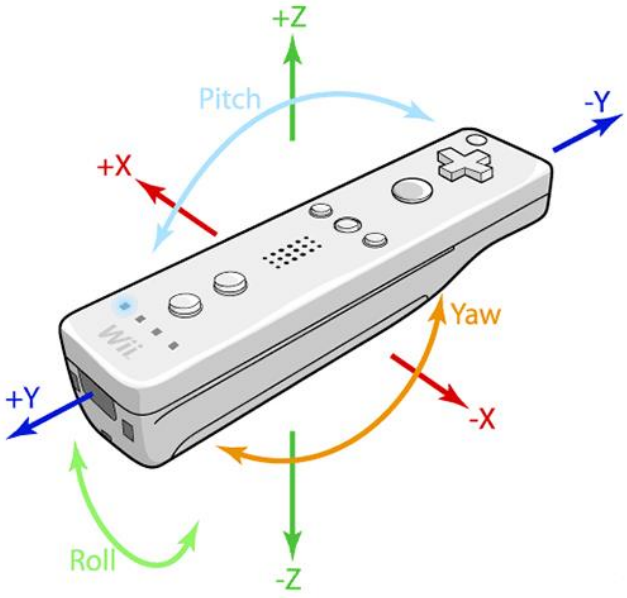


Ilustración 5. Ejes de Rotación

6.9.3 Tarea de desarrollo (T_2)

“Ya has conseguido configurar el entorno para programar prototipos de acelerómetro, tanto con Wiimote como con Arduino. Ahora, tu equipo y tu decidís indagar más en las posibilidades de interacción. Uno de tus compañeros ha desarrollado un pequeño Sketch de Processing donde se muestra un cubo girando en tres dimensiones. Para simular este giro utiliza primitivas que rotan la cámara con respecto a diferentes ejes. Tu objetivo será modificar esas primitivas para que hagan uso de los valores de inclinación (pitch) del acelerómetro (en Figura 5 un ejemplo de los ejes de rotación del acelerómetro relativos al dispositivo Wiimote).
Utilizando la información recogida en la primera tarea, decides probar a programar el prototipo, utilizando tanto drivers como la librería Simpleface, para evaluar cual se adapta mejor a las necesidades del desarrollo.
En el directorio donde está configurado el entorno de desarrollo encontrarás, junto con el sketch que ha preparado tu compañero, documentación tanto del driver como de la librería Simpleface.”

7 Resultados

7.1 Pre-Test

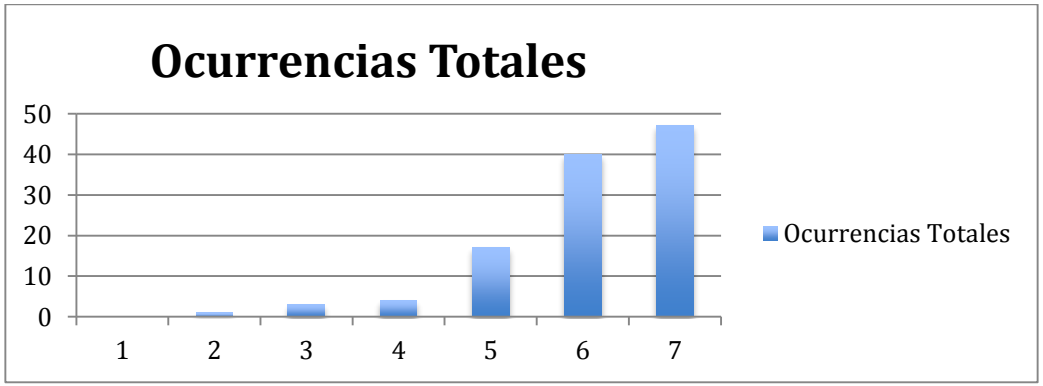
	18-24	**25-30**	**30+**					
Edad	4	11	1					
	Mujer	**Hombre**						
Sexo	4	12						
	NO	**1**	2	**3**	4	**5**	6	**7**
¿Conoce el lenguaje de programación Java? ¿Si es así, en qué medida?	0	0	0	1	1	7	7	0
¿Conoce el entorno de programación Processing? ¿Si es así, en qué medida?	15	0	1	0	0	0	0	0
	Si	**No**						
¿Ha programado alguna vez sistema que usen dispositivos de interacción natural?	2	14						
	Si	**No**						
De haber contestado si anteriormente ¿Eran dispositivos acelerómetros?	1	1						
	Si	**No**						
¿Están tus estudios relacionados con interfaces que hagan uso de dispositivos de interacción natural?	4	12						
		1	2	**3**	4	**5**	6	**7**
Cual consideras que es tu nivel de habilidad técnica a la hora de programar?		0	0	0	2	9	3	2

7.2 Pruebas

	Dispositivo	Orden	Tiempo Búsqueda	Tiempo Simpleface	Tiempo No Simpleface	Fallos Simpleface	Fallos No Simpleface	Ejercicio Completado (Con/Sin)
Sujeto 1	Arduino	Con/Sin	0:09:30	0:13:00	0:32:00	2	10	Si/Si
Sujeto 2	Arduino	Sin/Con	0:04:30	0:04:00	0:30:00	0	6	Si/Si
Sujeto 3	WiiMote	Con/Sin	0:07:00	0:09:00	0:17:30	1	1	Si/Si
Sujeto 4	WiiMote	Sin/con	0:04:00	0:13:00	0:15:00	0	0	No/No
Sujeto 5	WiiMote	Con/sin	0:03:17	0:14:30	0:32:00	1	5	Si/Si
Sujeto 6	Arduino	Con/Sin	0:07:00	0:20:00	0:40:00	3	5	Si/No
Sujeto 7	Arduino	Sin/Con	0:06:53	0:07:56	0:30:00	0	3	Si/Si
Sujeto 8	WiiMote	Sin/Con	0:04:30	0:07:56	0:31:22	1	5	Si/Si
Sujeto 9	Arduino	Con/Sin	0:10:56	0:40:00	0:40:00	3	3	No/No
Sujeto 10	WiiMote	Con/Sin	0:12:00	0:24:00	0:15:55	2	0	Si/Si
Sujeto 11	WiiMote	Sin/Con	0:06:40	0:15:00	0:34:00	0	4	Si/Si
Sujeto 12	Arduino	Sin/Con	0:10:27	0:14:48	0:40:00	3	4	Si/No
Sujeto 13	Arduino	Con/Sin	0:06:53	0:29:00	0:31:00	1	3	Si/Si
Sujeto 14	WiiMote	Con/Sin	0:06:50	0:08:00	0:20:37	1	6	Si/Si
Sujeto 15	WiiMote	Sin/Con	0:03:11	0:12:14	0:24:00	1	5	Si/Si
Sujeto 16	Arduino	Sin/Con	0:08:07	0:08:35	0:29:00	1	1	Si/Si
Total	**Arduino - 8**	**Sin 16**	**1:51:44**	**4:00:59**	**7:42:24**	**20**	**61**	**14(16)/12(16)**
	Wiimote - 8	**Con 16**						

7.3 Post-Test

	1	2	3	4	5	6	7
Estoy contento con lo fácil que ha sido usar el Simpleface	0	0	0	0	3	4	9
Ha sido fácil aprender a usar Simpleface	0	0	0	0	5	4	7
La librería podría aumentar mi productividad	0	0	0	0	1	9	6
La documentación que la acompaña ha sido clara	0	0	2	0	5	4	5
La información proporcionada me ha servido para completar las tareas de forma efectiva	0	1	0	0	2	8	5
El sistema tiene todas las funciones que esperaba que tuviera	0	0	1	3	1	4	7
En general, estoy satisfecho con el sistema	0	0	0	1	0	7	8
TOTAL	0	1	3	4	17	40	47



8 Interpretación estadística.

8.1 Software

Para todas las pruebas estadísticas realizadas se ha utilizado el software estadístico IBM SPSS Statics.

8.2 Prueba T de Student para muestras relacionadas

La prueba T de Student [Sealy, 1908] sirve para comparar las medias de dos variables cuantitativas, dado que todas las variables medidas durante la ejecución de la prueba lo son, se ha determinado que la prueba T de Student es la que mejor se ajusta a los propósitos del estudio. Esta prueba viene indicada para los casos en los cuales el tamaño muestra es demasiado pequeño como para que el estadístico en el que se basa este normalmente distribuido. Para suplir está carencia, se usa una estimación de la desviación típica en lugar del valor real. Además, la prueba se basa en comparación de medias lo cual se ajusta perfectamente a las hipótesis planteadas anteriormente, lo cual nos permite falsarlas o aceptarlas de forma sencilla mediante el análisis de los resultados de la prueba T.

Las observaciones seleccionadas en cada extracción se consideran relacionadas porque corresponden al mismo sujeto bajo dos tratamientos posibles, en este caso, con Simpleface y sin Simpleface.

Para poder aplicar este método el estudio debe cumplir las siguientes características:

- Asignación aleatoria de los grupos
- Homocedasticidad (homogeneidad de las varianzas de la variable dependiente de los grupos)
- Distribución normal de la variable dependiente en los dos grupos

La aplicación del test de t de Student exige el cumplimiento de supuestos de normalidad y Homocedasticidad. Lo primero significa que en las poblaciones las distribuciones de la variable que se compara debieran ser aproximadamente normales. Podemos suponer que nuestras variables serán distribuidas de forma

normal dado que los sujetos de estudio han sido seleccionados de manera aleatoria luego los tiempos y los errores procedentes de las pruebas serán aleatorios. Al igual que otros estadísticos como la altura o el coeficiente intelectual, nuestros estadísticos estarán distribuidos de forma normal, aun que debido al pequeño tamaño de la muestra no se pueda apreciar gráficamente. Además de esto, podemos comprobar que existe poca diferencia entre la media y la mediana de nuestras muestras. Con respecto a la homocedasticidad, significa que las varianzas en ambos grupos (Con y sin simpleface), deben ser similares. La ausencia de estás dos precondiciones no hace imposible la aplicación de la T de Student, pero sí que hace a sus resultados perder robustez

A continuación se muestran los datos que corroboran ambos supuestos.

Tiempo No Simpleface	Media	Mediana	Varianza	Fallos No Simpleface	Media	Mediana	Varianza
0:32:00	0:28:54	0:30:30	3,05069E-05	10	3,8125	4	6,69583333
0:30:00				6			
0:17:30				1			
0:15:00				0			
0:32:00				5			
0:40:00				5			
0:30:00				3			
0:31:22				5			
0:40:00				3			
0:15:55				0			
0:34:00				4			
0:40:00				4			
0:31:00				3			
0:20:37				6			
0:24:00				5			
0:29:00				1			

Tiempo Simpleface	Media	Mediana	Varianza	Fallos Simpleface	Media	Mediana	Varianza
0:13:00	0:15:04	0:13:00	3,88796E-05	2	1,25	1	1,0625
0:04:00				0			
0:09:00				1			
0:13:00				0			
0:14:30				1			
0:20:00				3			
0:07:56				0			
0:07:56				1			
0:40:00				3			
0:24:00				2			
0:15:00				0			
0:14:48				3			
0:29:00				1			
0:08:00				1			
0:12:14				1			
0:08:35				1			

8.2.1 Prueba T - Pueba T1

Tenemos dos conjuntos de tiempos que se han obtenido al medir dos veces en los mismos sujetos (muestras relacionadas) dos variables cuantitativas (Tiempo de búsqueda con Simpleface y sin Simpleface)

Hipótesis:

H_0: $\mu_{BúsquedaConSimpleface} \geq \mu_{BúsquedaSinSimpleface}$

H_1: $\mu_{BúsquedaConSimpleface} < \mu_{BúsquedaSinSimpleface}$

(contraste unilateral izquierdo)

Estadístico de contraste:

Estadísticos de muestras relacionadas

		Media	N	Desviación típ.	Error típ. de la media
Par 1	Tiempo_búsqueda_SinSimpleface	6,9825	16	2,70733	,67683
	Tiempo_búsqueda_ConSimpleface	,00	16	,000	,000

Correlaciones de muestras relacionadas

		N	Correlación	Sig.
Par 1	Tiempo_búsqueda_SinSimpleface y Tiempo_búsqueda_ConSimpleface	16	.	.

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia				
					Inferior	Superior			
Par 1	Tiempo_búsqueda_SinSimpleface - Tiempo_búsqueda_ConSimpleface	6,98250	2,70733	,67683	5,53987	8,42513	10,316	15	,000

Podemos comprobar con un nivel de confianza del 95% que la diferencia entre el tiempo de búsqueda con Simpleface y el tiempo de búsqueda sin Simpleface se encuentra entre 5,53987 y 8,42513 puntos. Las tres últimas columnas informan sobre el valor del estadístico t, sus grado de libertad (gl) y el nivel crítico bilateral (el unilateral se obtiene dividiendo el bilateral entre dos, en este caso sería el mismo 0,000). Puesto que el valor del nivel crítico es menor que 0,05 rechazamos la hipótesis de igualdad de medias y concluimos que la media en el tiempo de búsqueda sin Simpleface es mayor que la media obtenida con Simpleface y, por tanto, podemos concluir que Simpleface minimiza la cantidad de conocimientos previos necesarios para la configuración del entorno de desarrollo.

8.2.2 Prueba T - Prueba T2 (Tiempo)

Comparación de medias de tiempo con Simpleface y sin Simpleface.

Hipótesis: $H_0: \mu_{ConSimpleface} \geq \mu_{SinSimpleface}$

$H_1: \mu_{ConSimpleface} < \mu_{SinSimpleface}$ (contraste unilateral izquierdo)

Estadísticos de muestras relacionadas

		Media	N	Desviación típ.	Error típ. de la media
Par 1	Tiempo_simpleface	15,0606	16	9,27394	2,31849
	Tiempo_No_simpleface	28,9006	16	8,21389	2,05347

Correlaciones de muestras relacionadas

		N	Correlación	Sig.
Par 1	Tiempo_simpleface y Tiempo_No_simpleface	16	,320	,227

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia				
					Inferior	Superior			
Par 1	Tiempo_simpleface - Tiempo_No_simpleface	-13,84000	10,23520	2,55880	-19,29395	-8,38605	-5,409	15	,000

Podemos comprobar con un nivel de confianza del 95% que la diferencia entre el tiempo con Simpleface y el tiempo sin Simpleface se encuentra entre -19,29395 y -8,38605 puntos. El nivel crítico unilateral se obtiene dividiendo el bilateral entre dos, en este caso sería el mismo 0,000. Puesto que el valor del nivel crítico es menor que 0,05 rechazamos la hipótesis de igualdad de medias y concluimos que la media en el tiempo sin Simpleface es mayor que la media obtenida con Simpleface, Por lo que el tiempo en realizar la tarea con Simplace es significativamente menor y por tanto, podríamos decir que aumentamos la eficacia de manera positiva usando la herramienta que sin ella.

8.2.3 Prueba T - Prueba T2 (Fallos)

Comparación medias fallos con Simpleface y sin Simpleface.

Hipótesis: H_0: $\mu_{FallosConSimpleface} \geq \mu_{FallosSinSimpleface}$

H_1: $\mu_{FallosConSimpleface} < \mu_{FallosSinSimpleface}$ (contraste unilateral izquierdo)

Estadísticos de muestras relacionadas

		Media	N	Desviación típ.	Error típ. de la media
Par 1	Fallos_Simpleface	1,25	16	1,065	,266
	Fallos_No_Simpleface	3,81	16	2,588	,647

Correlaciones de muestras relacionadas

		N	Correlación	Sig.
Par 1	Fallos_Simpleface y Fallos_No_Simpleface	16	,139	,607

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia				
					Inferior	Superior			
Par 1	Fallos_Simpleface - Fallos_No_Simpleface	-2,563	2,658	,664	-3,979	-1,146	-3,857	15	,002

Podemos comprobar con un nivel de confianza del 95% que la diferencia entre el tiempo con Simpleface y el tiempo sin Simpleface se encuentra entre -3,979 y -1,146 puntos. El nivel crítico unilateral se obtiene dividiendo el bilateral entre dos, en este caso sería el mismo 0,001. Puesto que el valor del nivel crítico es menor que 0,05 rechazamos la hipótesis de igualdad de medias y concluimos que la media de fallos sin Simpleface es mayor que los fallos obtenidos con Simpleface, Por lo que el número de fallos producidos al realizar la tarea con Simplace es significativamente menor y por tanto, podríamos decir que aumentamos la eficacia de manera positiva usando la herramienta que sin ella.

8.3 Porcentajes Prueba Terminada con Simpleface y sin Simpleface:

Ejercicio completado con Simpleface

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	0	2	12,5	12,5	12,5
	1	14	87,5	87,5	100,0
	Total	16	100,0	100,0	

Ejercicio completado sin Simpleface

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	0	4	25,0	25,0	25,0
	1	12	75,0	75,0	100,0
	Total	16	100,0	100,0	

Como podemos observar, el porcentaje de ejercicios completados con existe usando Simpleface, fue del 87,5%, mientras que el porcentaje de éxito sin Simpleface fue tan solo de un 75%. Podemos concluir que Simpleface consiguió 12,5 puntos más en porcentaje de éxito, sumado a las conclusiones extraídas de las hipótesis.

8.4 Conclusiones estadísticas.

Basándonos en el análisis anteriormente citado, podemos negar las hipótesis nulas, y por lo tanto, afirmar las hipótesis alternativas, con lo cual, podemos concluir con significancia estadística que:

- La herramienta Simpleface minimiza la cantidad de conocimientos previos necesarios para la configuración del entorno desarrollo.

- La herramienta Simpleface afecta de manera positiva a la eficiencia (tiempo de desarrollo, cantidad de errores por línea) en el desarrollo de los prototipos que hagan uso de dispositivos para la interacción natural.

8.5 Conclusiones cualitativas

Basándonos en los test previos y posteriores, así como en las notas tomadas durante la prueba (recordemos el protocolo Think Aloud), podemos sacar una serie de conclusiones de la prueba.

- La documentación de Simpleface debe mejorarse. A muchos participantes, a la hora de usar las funciones que ofrece Simpleface, les pareció confusa la descripción de las mismas y dejaron constancia, tanto en el test posterior como verbalmente durante la prueba, que debía ser un aspecto a mejorar. Así mismo, también creyeron que hacen falta más ejemplos para completar la documentación de las funciones. Esta ha sido la conclusión que mas participantes han comentado.

- Generalmente, el enunciado a de la prueba ha sido comprendido, sin embargo, varios participantes han presentado problemas los conceptos de aceleración medida con los acelerómetros, así como los ejes de giro. Varios participantes dejaron constancia que les costó diferencia entre inclinación y aceleración en la librería Simpleface. Este comportamiento es esperado ya que en los test previos solo un sujeto ha indicado haber utilizado acelerómetros previamente. De cualquier forma, esta carencia ha de ser cubierto por una mejor documentación, como se ha comentado en el punto anterior.

- El perfil de los sujetos de estudio fue correcto para el desarrollo de la prueba. Todos ellos conocían los principios básicos exigidos para la realización, programación orientada a objetos y más concretamente, Java. Además, en términos generales ha tenido una curva de aprendizaje similar, dando a

entender que su perfil homogéneo en cuanto a estudios ha beneficiado la realización del estudio.

- Muchos de los participantes sugirieron posibles ampliaciones del sistema, tales como nuevas funcionalidades o extensiones sobre las ya descritas, ya sea para corregir fallos o mejorar las capacidades de Simpleface.

- La tarea que utiliza el Wiimote como dispositivo ha resultado ser, en términos generales, más sencilla que el Arduino cuando no se usaba Simpleface. El manejo del bus serie ha dificultado la realización.

- Los participantes estiman, de media, que tienen una habilidad a la hora de programa de 5.3 sobre 7, lo cual se ajusta a los resultados obtenidos.

9 Gestión del proyecto.

9.1 Modelos

Dado que el proyecto Simpleface no se ajusta al desarrollo de un producto típico del entorno empresarial, la elección del modelo de desarrollo para la creación de la prueba de concepto es trivial. En cualquier caso, se han aplicado las etapas de recogida de requisitos, (más como un compromiso con el experimento que como requisitos del software en sí), diseño e implementación y pruebas pertenecientes al modelo de cascada. Debido a la simplicidad de la implementación no se han requerido iteraciones.

9.2 Gestión del proyecto

A continuación se muestra el gráfico GANTT con la planificación del proyecto, incluyendo el desarrollo y la evaluación experimental.

Name	Duration	Start	Finish
Introducción	**8 days?**	**2/1/12 8:00 AM**	**2/10/12 5:00 PM**
Planteamiento PFC	3 days	2/1/12 8:00 AM	2/3/12 5:00 PM
Estado del arte	5 days?	2/6/12 8:00 AM	2/10/12 5:00 PM
Analisis	**4 days?**	**2/13/12 8:00 AM**	**2/16/12 5:00 PM**
Requisitos	2 days?	2/13/12 8:00 AM	2/14/12 5:00 PM
Casos de uso	2 days?	2/15/12 8:00 AM	2/16/12 5:00 PM
Diseño	**5 days**	**2/17/12 8:00 AM**	**2/23/12 5:00 PM**
Arquitectura	5 days	2/17/12 8:00 AM	2/23/12 5:00 PM
Implementación	**9 days**	**2/24/12 8:00 AM**	**3/7/12 5:00 PM**
Estandar Acelerometro	4 days	2/24/12 8:00 AM	2/29/12 5:00 PM
Modulo Arduino	3 days	3/1/12 8:00 AM	3/5/12 5:00 PM
Modulo WiiMote	2 days	3/6/12 8:00 AM	3/7/12 5:00 PM
Evaluación	**15 days?**	**3/8/12 8:00 AM**	**3/28/12 5:00 PM**
Diseño	5 days?	3/8/12 8:00 AM	3/14/12 5:00 PM
Evaluación experimental	5 days	3/15/12 8:00 AM	3/21/12 5:00 PM
Interpretación de los res...	5 days?	3/22/12 8:00 AM	3/28/12 5:00 PM
Memoria	**27 days?**	**3/29/12 8:00 AM**	**5/4/12 5:00 PM**
Escritura	20 days	3/29/12 8:00 AM	4/25/12 5:00 PM
Correcciones	5 days?	4/26/12 8:00 AM	5/2/12 5:00 PM
Presentación	2 days	5/3/12 8:00 AM	5/4/12 5:00 PM

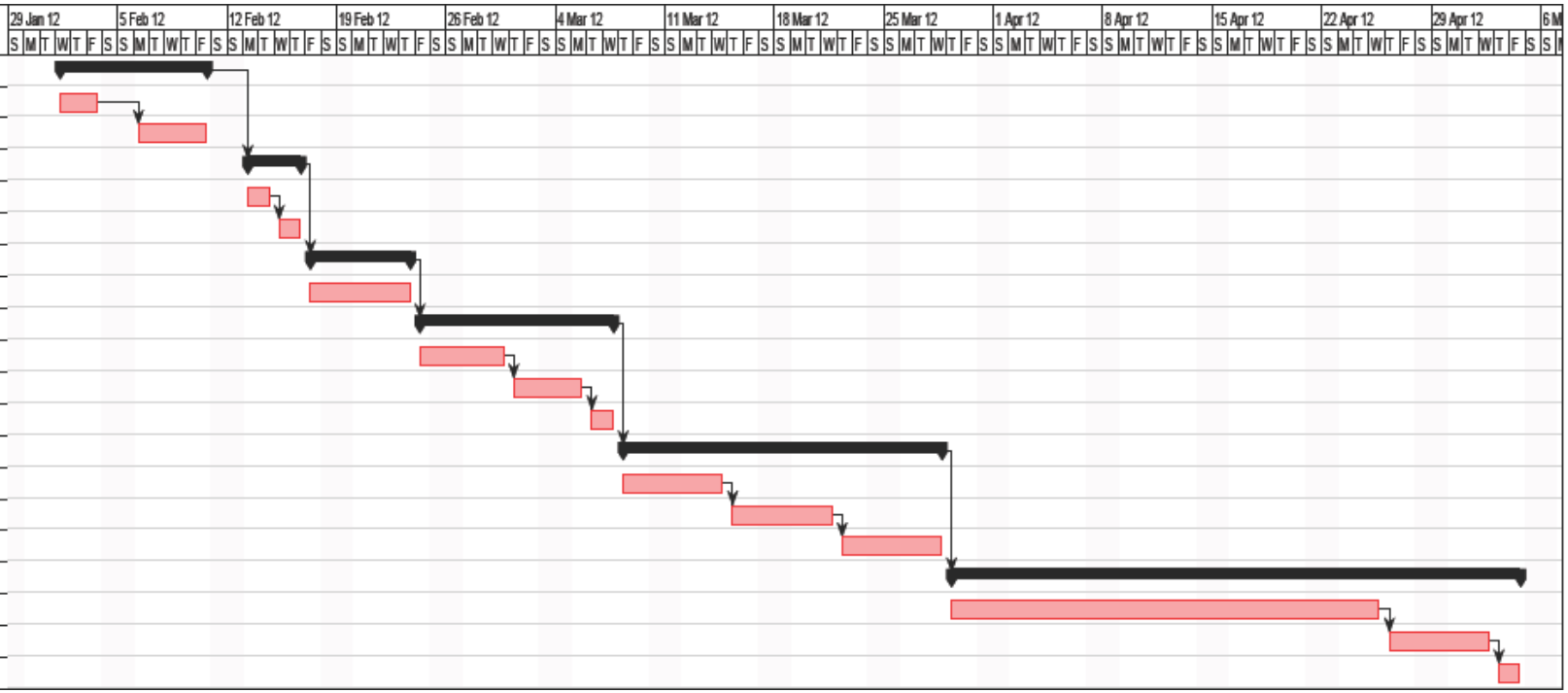


Ilustración 6 Diagrama GANTT

10 Conclusiones y trabajos futuros

10.1 Conclusiones

El desarrollo y posterior evaluación de la herramienta Simpleface ha resultado satisfacer los criterios para los que fuera creado, mejorar la eficiencia en el desarrollo de aplicaciones que hagan uso de dispositivos de interacción natural y minimizar los conocimientos necesarios para la creación de prototipos. Además de las evidencias estadísticas, la evaluación de la herramienta a permitido concluir diversas mejoras para el sistema, así como la metodología en próximas evaluaciones.

En líneas generales, Simpleface ha demostrado ser una prueba de concepto de lo que puede llegar a ser un herramienta accesible, modular y fácilmente ampliable para el desarrollo de prototipos que hagan uso de dispositivos de interfaz, los cuales, cada vez serán más comunes en nuestra sociedad. Este proyecto ha sido llevado a cabo con el convencimiento de que al acercar el desarrollo de prototipos a personas que están más alejadas de la técnica informática, se crearan mas y mejores ideas, pues solo cuando las personas pueden dar forma a lo que tienen dentro de sus cabezas, se produce innovación.

10.2 Trabajos futuros

Como se ha mencionado, Simpleface ha sido creado como una prueba de concepto para evaluar una teoría, lo cual deja muchísimo margen al desarrollo de la plataforma.
Comenzando por lo más obvio, el soporte de nuevos dispositivos de interacción natural, hasta lo más complejo, ser portado a diferentes lenguajes fuera del entorno Processing, Simpleface puede crecer como una plataforma abierta si consigue aunar el suficiente interés por parte de la comunidad de desarrolladores.

11 Referencias

Johnson, T. (1969). Sketchpad III: a computer program for drawing in three dimensions.

Weiser, M. (1991). The Computer for the 21st Century.

Weiser, M. (1993). Some computer science issues in ubiquitous computing.

Satyanarayanan, M. (2001) Pervasive computing: vision and challenges.

Sasha D. and Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century.

Myers, B., Hudson, S. E., and Pausch, R. (2000). Past, present, and future of user interface software tools.

Klemmer, S. (2012). The power of prototyping.

York, J. and Pendharkar, P. (2004).Human–computer interaction issues for mobile computing in a variable work context.

Davidson, P. and Han, J. (2006). Synthesis and control on large scale multi-touch sensing displays.

Mistry, P., Maes, P. and Chang, L. (2009). WUW - wear Ur world: a wearable gestural interface.

Kuznetsov, S. and Paulos, E. (2010). Rise of the expert amateur: DIY projects, communities, and cultures.

Lim, Y.-K., Stolterman, E., and Tenenberg, J. (2008). The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas.

Buxton, B. (2007). Sketching User Experiences: Getting the Design Right and the Right Design.

Ericsson, K. and Simon, H. (1980). Verbal reports as data.

12 ANEXO A

Encuesta Preliminar.

1. Edad

a) 18-24
b) 25-30
c) 30+

2. Sexo

a) Mujer
b) Hombre

¿Conoce el lenguaje de programación Java (Paradigma, sintaxis, características, etc)?

a) Si
b) No

7.2. Si ha contestado "Si" a la anterior pregunta. ¿Con que frecuencia lo usa?

a) Nunca
b) Esporádicamente
c) Habitualmente

¿Conoce el entorno de desarrollo Processing (Características, etc)?

a) Si
b) No

8.2. Si ha contestado "Si" a la anterior pregunta. ¿Con que frecuencia lo usa?

a) Nunca
b) Esporádicamente
c) Habitualmente

¿Ha programado alguna vez sistemas que utilicen dispositivos para interacción natural?

a) Si
b) No

Si ha contestado "Si" a la anterior pregunta. ¿Han sido acelerómetros algunos de esos dispositivos?

a) Si
b) No

¿Están sus estudios relacionados con las interfaces de usuario que utilizan dispositivos para interacción natural?

a) Si
b) No

Siendo 1 el mínimo y 7 el máximo. ¿Cuál considera que es su nivel de habilidad técnica a la hora de programar?

1 2 3 4 5 6 7

13 ANEXO B

Encuesta post-estudio.

1. En general, estoy contento con lo fácil que ha sido utilizar Simpleface.

1 2 3 4 5 6 7

2. Ha sido fácil aprender a usar Simpleface.

1 2 3 4 5 6 7

3. Creo la librería podría aumentar mi productividad.

1 2 3 4 5 6 7

4. La documentación que acompaña el sistema ha sido clara.

1 2 3 4 5 6 7

5. La información proporcionada me ha servido para completar las tareas de forma efectiva.

1 2 3 4 5 6 7

6. El sistema tiene todas las funciones que esperaba que tuviera.

1 2 3 4 5 6 7

7. En general, estoy satisfecho con el sistema.

1 2 3 4 5 6 7

8. Comentarios sobre la librería. Sugerencias, funcionalidades que te gustaría que incorporase, etc.

_____
_____
_____
_____
_____
_____
_____